De novo fragment assembly with short mate-paired reads: Does the read length matter?

Mark J. Chaisson,[1,3] Dumitru Brinza,[2] and Pavel A. Pevzner[2]

[1]*Bioinformatics Program, University of California San Diego, La Jolla, California 92093, USA;* [2]*Department of Computer Science and Engineering, University of California San Diego, La Jolla, California 92093, USA*

Increasing read length is currently viewed as the crucial condition for fragment assembly with next-generation sequencing technologies. However, introducing mate-paired reads (separated by a gap of length, GapLength) opens a possibility to transform short mate-pairs into long mate-reads of length ≈ GapLength, and thus raises the question as to whether the read length (as opposed to GapLength) even matters. We describe a new tool, EULER-USR, for assembling mate-paired short reads and use it to analyze the question of whether the read length matters. We further complement the ongoing experimental efforts to maximize read length by a new computational approach for increasing the effective read length. While the common practice is to trim the error-prone tails of the reads, we present an approach that substitutes trimming with error correction using repeat graphs. An important and counterintuitive implication of this result is that one may extend sequencing reactions that degrade with length "past their prime" to where the error rate grows above what is normally acceptable for fragment assembly.

[Supplemental material is available online at www.genome.org.]

The field of high-throughput sequencing has grown recently in both applications and computational support. This is enabled by the many platforms that exist for high-throughput sequencing, including those produced by 454 Life Sciences (Roche) (Margulies and Egholm 2005), Illumina 1G Genome Analysis System (www.illumina.com), Applied Biosystems SOLiD Sequencing (www.appliedbiosystems.com), and Helicos GSS Sequencing (www.helicosbio.com). Although the 454 sequencing platform is now producing reads that are of similar length to Sanger reads, the underlying paradigm is that a higher throughput may be achieved at the sacrifice of read length. The technologies with the highest throughput currently available produce short, 20–40 base reads, distinguished as ultrashort reads.

Many recent successful applications of ultrashort reads have used the reference genomes for whole-genome resequencing (Hillier et al. 2008), chromatin remodeling mapping (Schones and Zhao 2008), and whole-genome methylation (Barsky et al. 2007) profiling. An analysis in Whiteford et al. (2005) showed that the number of reads uniquely mapped to the human genome grows with the increase in read length, but reaches a plateau after the first ≈40 nt (nucleotides). This implies that there is little incentive to increase the read length in resequencing applications, since it provides little return on investment. While generating 40-nt reads is usually sufficient for resequencing, de novo fragment assembly may require longer reads. The Illumina platform can easily generate longer reads, but the error rates deteriorate after the first 35 nt, making the ends of reads not suitable for fragment assembly. This again provides little incentive to generate longer reads. By extending the effective length of reads, our EULER-USR assembler generates more reads that span the repeats, and thus improves the assemblies.

Most de novo assemblers for Sanger reads follow the "overlap-layout-consensus" paradigm that is optimized for such reads (Huang et al. 2003; Jaffe et al. 2003) and does not scale well for short-read assembly. In contrast, most approaches to short-read assembly including EULER-SR (Chaisson and Pevzer 2008), Velvet (Zerbino and Birney 2008), and ALLPATHS (Butler et al. 2008) use an alternative Eulerian approach that model the assembly problem as a search for a Eulerian path in a de Bruijn graph (Pevzner et al. 2001).

The advantage of the Eulerian approach is that it generates a theoretically optimal assembly of reads of length k (in high-coverage projects) by essentially mimicking fragment assembly as a Sequencing by Hybridization (SBH) problem on a virtual DNA array with all k-mers (Pevzner 1989). Idury and Waterman (1995) demonstrated how the Eulerian approach for SBH can be applied to fragment assembly of Sanger reads. However, the Eulerian approach works best for error-free reads and quickly deteriorates as soon as the reads have even a small number of base-calling errors. To alleviate this limitation, two different error-correction approaches are used: error correction in reads prior to assembly (Pevzner et al. 2001; Butler et al. 2008), and post-hoc graph corrections that remove spurious edges from the assembled de Bruijn or A-Bruijn graphs (Pevzner et al. 2004; Zerbino and Birney 2008).

Correcting errors in reads prior to assembly was shown to be useful for both Sanger and 454 reads (Pevzner et al. 2001; Chaisson and Pevzner 2008) (error rates reduced 40-fold from 1.2% to 0.03% for Sanger reads). However, this approach essentially transforms already rather accurate reads (≈1% error rate) into nearly error-free reads. The efficiency of the existing error correction approaches quickly deteriorates with even a small increase in the error rates of original reads (e.g., from 1% to 3%). For example, an optimized error correction tool from Tammi et al. (2003) was able to reduce the error rates from 3.4% to 0.3% on simulated data, only a 10-fold decrease. While it looks like a reasonably low error rate, it makes it nearly impossible to apply the Eulerian approach that does not tolerate even such seemingly small error rates. This makes the full-length Illumina reads "ineligible" for Eulerian assembly (Fig. 1 illustrates that the error rate is ≈20% at the ends of reads) and sets an accuracy bottleneck for developing new sequencing technologies.

In this work we focus on the Illumina technology and describe how to increase the "usable" length of error-prone Illumina reads while keeping them nearly error-free. Although this study

[3]Corresponding author.
E-mail mchaisso@bioinf.ucsd.edu; fax (858) 534-7029.
Article published online before print. Article and publication date are at http://www.genome.org/cgi/doi/10.1101/gr.079053.108.

19:336–346 © 2009 by Cold Spring Harbor Laboratory Press; ISSN 1088-9051/09; www.genome.org

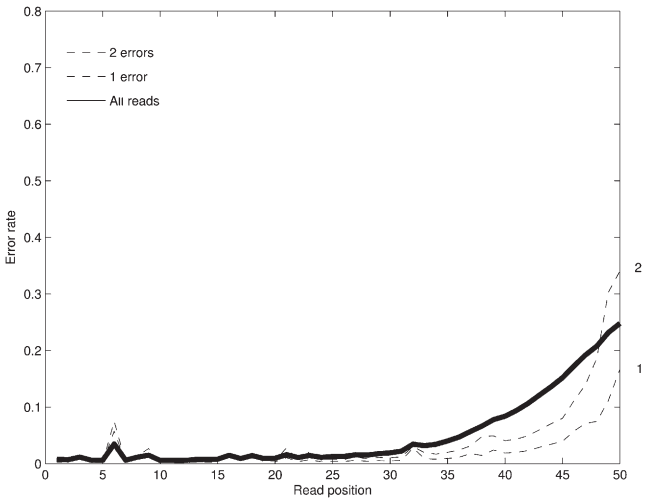

Figure 1. The positional profile of base-calling errors for Illumina reads for 2 million 50-nt-long reads from a human BAC. The error rate across reads is shown (solid line) along with the error rate for reads with a fixed number of errors. The erroneous nucleotides in each read are detected by mapping the read to the reference genome. The high error rate in position 6 is due to the bias in our particular data set rather than a systematic problem with the Illumina technology.

limits the benchmarking of EULER-USR to the Illumina technology, our algorithm is applicable to any technology characterized by high error rates, such as the Helicos platform. While the average error rate in Illumina reads is under 2% for the first ≈30 nt, it quickly increases in the tails of the reads reaching ≈20% at position 50. No existing short-read assembly tool can efficiently deal with such high error rates, and the conventional wisdom is that the read tails become unusable when the error rate exceeds 3%. Below we introduce an alternative error-correction approach that uses the de Bruijn graph constructed on the accurate read prefixes in order to correct the error-prone read suffixes by fitting them into the de Bruijn graph. Since EULER-USR can assemble error-prone reads, we hope that it can catalyze developments of sequencing platforms aimed at generating longer but less accurate sequencing reads.

Reads may be combined with mate-paired information to further improve the quality of assemblies, and the next generation of sequencing companies are actively pursuing both increasing the read length and effectively generating mate-pairs. For example, Illumina recently increased the effective read length from 35 to 50 nt and announced plans to provide capabilities for 75–100 nt reads in 2009. On the other hand, Illumina and various sequencing centers are exploring applications of jump and PET libraries (Ng et al. 2006) for generating mate-pairs in the context of short-read technologies. While increasing the read length is a high priority for most next-generation sequencing companies, there exists an opinion that the read length almost does not matter if one uses mate-paired reads. Indeed, Pevzner and Tang (2001) demonstrated that most mate-pairs: "read$_{start}$-GAP of length d-read$_{end}$" can be transformed into mate-reads: "read$_{start}$-SEQUENCE of length d-read$_{end}$" by filling in the gap of length d with the nucleotide sequence representing an appropriate path in the de Bruijn graph. As a result, one can generate contiguous long reads of length $2 \cdot l + d$ (span of mate pairs) from short mate-paired reads of length l, making the read length almost irrelevant (typically, $d >> l$).

We show how EULER-USR utilizes mate pairs to significantly improve assembly, and further use it to answer the question of whether read length matters. We demonstrate that the answer to this question is closely linked with the efficiency of transforming mate pairs into mate-reads (the percentage of mate pairs successfully transformed into mate-reads). When the read length exceeds a certain threshold, the read-length barrier, the efficiency reaches nearly 100%, so that the read length, indeed, does not matter. For example, for the *Escherichia coli* genome, the read-length barrier is ≈35 nt.[4] This is good news for technologies with reads already longer than 35 nt (e.g., Illumina reads) but bad news for technologies with shorter reads (e.g., Helicos and SOLiD reads). However, while the current parameters of Illumina reads may be already sufficient for reliable assembly of some bacterial genomes, they are not sufficient for slightly larger genomes like *Saccharomyces cerevisiae* with higher read-length barriers. This observation reveals a synergy between EULER-USR error-correction approach to increasing the read length and EULER-USR approach to transforming mate pairs into mate-reads. Indeed, while the length of the "usable" portions of Illumina reads is currently below the read-length barrier for yeast, EULER-USR error correction allows one to increase the effective read length beyond the read-length barrier. Therefore, while the mate-paired information represents by far the most important factor for improving the assembly quality, the read length also provides a valuable contribution to the assembly.

The EULER-USR software for assembling mate-paired short reads is publicly available at http://euler-assembler.ucsd.edu.

Methods

We have developed the EULER-USR algorithm for assembling mate-paired and error-prone ultrashort reads. In addition to the previous Eulerian approaches (Pevzner et al. 2001) that correct reads based on k-mer multiplicities, EULER-USR corrects reads based on how they map to repeat graphs (Pevzner et al. 2001; Chaisson and Pevzner 2008), a generalization of the de Bruijn graphs.

The de Bruijn graph of a genome is constructed on the set of all k-mers in the genome. Vertices u and v are connected by a directed edge (u, v) if there is a $(k + 1)$-mer in the genome that has the k-mer u as a prefix and the k-mer v as a suffix. A small example of a de Bruijn graph is shown in Figure 2A,B. As a result of this construction, every substring of length $>k$ in the genome maps to a unique path in its de Bruijn graph. Similar to the de Bruijn graph of a genome (Fig. 2C), one can construct the de Bruijn graph of reads (Fig. 2D) on the set of all k-mers present in reads. The de Bruijn graphs of real genomes are very complex, a reflection of a large number of repeats with slightly varying repeat copies. A repeat graph of a genome (or reads) is a "simplified" version of the de Bruijn graph with small bulges and whirls removed (Fig. 2E,F; Pevzner et al. 2004; Myers 2005). The key observation in the Eulerian assembly is that the repeat graph of a genome can be approximated by the repeat graph of reads, and thus may be constructed from reads alone, i.e., without knowing the genome (Fig. 2, cf. E and F; Pevzner et al. 2004).

If the repeat graph of the genome is known, one may correct errors in a read by simply mapping this read to a path in the repeat graph and substituting the read by the path.[5] While this procedure would result in a nearly error-free set of reads, it is not clear how to construct a repeat graph from inaccurate reads and how to further

[4]We emphasize that the read-length barrier depends on the genome, the span of mate-pairs, coverage, error-rates in reads, variability in gap length, etc. The read-length barrier ≈35 nt for *E. coli* was computed under the assumption that the span is 300 ± 30 nt.

[5]This procedure may also result in error corruption (Pevzner et al. 2001).

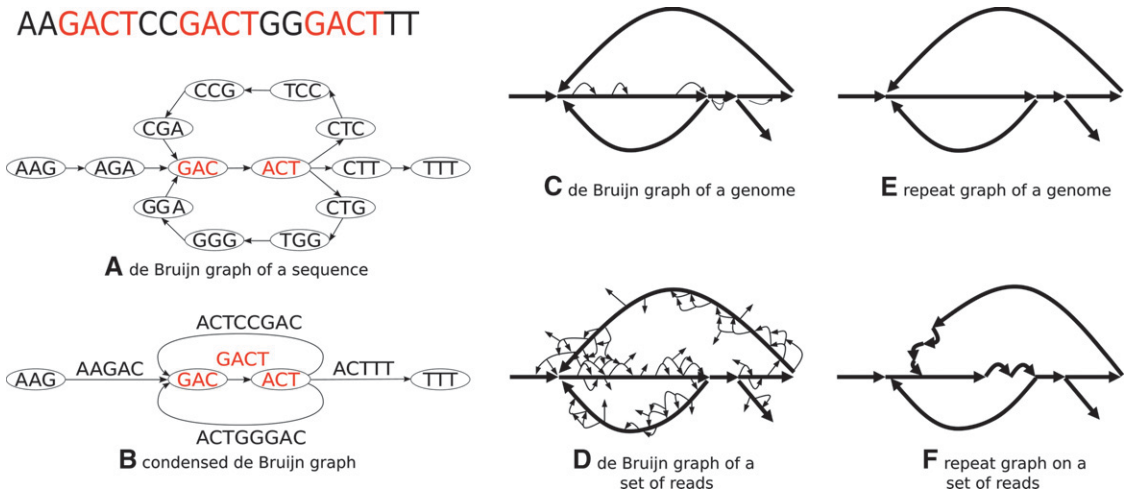

Figure 2. From de Bruijn graphs to repeat graphs. The de Bruijn graph of a sequence contains a vertex for every *k*-mer in the sequence, and an edge (*u, v*) for every pair of consecutive (overlapping) *k*-mers in the sequence (*A*). The condensed de Bruijn graph replaces all paths containing nonbranching vertices by a single edge labeled by the sequence that generated the path (*B*). When the condensed de Bruijn graph is constructed on a genome, it contains some small bulges and whirls representing repeats with slightly varying repeat copies (*C*). In the repeat graph, the bulges and whirls are removed (*E*). The de Bruijn graph of reads contains additional spurious bulges and whirls caused by sequencing errors in reads (*D*). The goal of the Eulerian assembly is to construct the repeat graph of reads (*F*) that approximates the repeat graph of the genome. Different studies use different terminology, e.g., the edges of these graphs are referred to as "blocks" in Zerbino and Birney (2008) and "unipaths" in Butler et al. (2008).

map such reads to the repeat graph. In our approach, we construct the repeat graph from (accurate) read prefixes and then map entire reads (with inaccurate suffixes) to this graph by threading.

The EULER-USR algorithm consists of the following three steps that can be further supplemented by the threading procedure that we will describe later (see Assembling error-prone reads [error-correction by threading]):

1. Detecting accurate read prefixes and correcting errors within them using frequent *k*-mers. This operation generates the set of extremely accurate (nearly error-free) read prefixes.
2. Constructing the repeat graph on error-corrected prefixes using *k*-mers.
3. Simplifying the repeat graph after transforming mate-pairs into mate-reads.

Detecting and error correcting accurate read prefixes

Although the quality of Illumina reads deteriorates with length, the prefixes are quite accurate (<2% error rate). While many reads can be turned into error-free reads by our error-correction algorithm, errors will remain in low-quality reads even after error correction. In this case, it is important to detect the longest read prefix that may be error corrected and discard the reads that cannot be corrected.

We correct reads using a modified version of the Spectral Alignment (SA) algorithm described in Pevzner et al. (2001) and Chaisson et al. (2004). The SA (Spectrum, read) method corrects read given a set of *k*-mers Spectrum. Given a set of reads *R* and a frequency threshold *m*, we define a spectrum [Spectrum = $\text{Spectrum}_k(R, m)$] as the set of all *k*-mers that appear at least *m* times in reads from *R*. The set of such solid *k*-mers approximates the set of all *k*-mers in the genome. We define a read as error free if all of its *k*-mers are solid; otherwise, we attempt to make a read error free by mutating a few nucleotides in the read (Pevzner et al. 2001; Chaisson and Pevzner 2008). We only consider substitution mutations since there are few insertions/deletions in Illumina reads.

We use a greedy heuristic to find the minimum number of mutations to make every *k*-mer in a read solid. It records the number of *k*-mers that are made solid for all three possible mutations at every position in the read. The mutation that makes the highest number of *k*-mers in the read solid is applied if it "solidifies" more than *t* *k*-mers, where *t* is an internal threshold (Chaisson and Pevzner 2008). This heuristic is iteratively applied either until all *k*-mers in the read become solid, or until there are no mutations that can solidify more than *t* *k*-mers. We output the prefix of the read that is solid, or discard the read if none of it is solid. This partitions all reads that are not discarded into a prefix (accurate) and suffix (less accurate) that will be corrected at a later stage. The set of reads is divided into two partitions: fixed and unfixable. The fixed partition has reads that have a solid prefix, and the unfixable partition contains reads with no solid *k*-mers. A similar method is used in Butler et al. (2008).

In order to choose the multiplicity threshold *m*, we assume that *k*-mers are generated from a mixture of two models: one erroneous and the other correct (Fig. 3). If we assume positional independence of errors in reads, the multiplicity of erroneous *k*-mers follows a Poisson distribution, and the multiplicity of correct *k*-mers follows a Poisson with a large mean that may be approximated by a Gaussian. We choose *m* by fitting a Poisson and Gaussian mixture model to the distribution of *k*-mer multiplicity and finding the first local minimum of this distribution (Fig. 3).

Constructing the repeat graph on error-corrected read prefixes

We construct the de Bruijn graph on the error-corrected read prefixes and further transform it into the repeat graph. Reads that contain errors or SNPs in the middle create short undirected cycles in the de Bruijn graph called bulges, and reads that contain errors in the end create erroneous sources or sinks. Finally, some read errors form chimeric reads by transforming the sequence in one end of the read to that of a distant part of the genome, creating an edge that erroneously connects two unrelated contigs. Transformation of the de Bruijn graph into the repeat graph amounts to

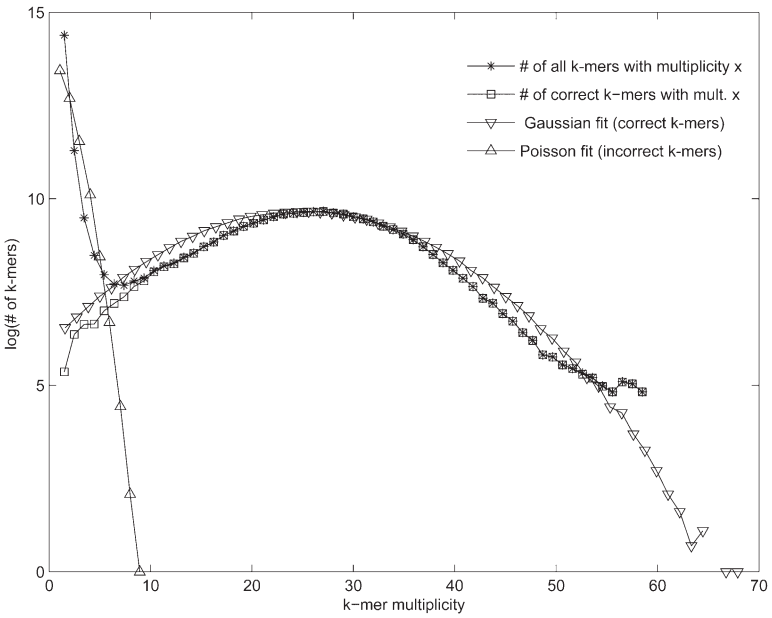

Figure 3. Choosing the multiplicity threshold for error correction. All *k*-mers appearing in the reads are classified as correct if they appear in the genome, and incorrect otherwise. For a multiplicity *x*, let correct(*x*)/incorrect(*x*) be the number of correct/incorrect *k*-mers with multiplicity *x* (the plots are shown for 50-base long Illumina reads from a human BAC and $k = 20$). As expected, most high-multiplicity *k*-mers are correct and most low-multiplicity *k*-mers are incorrect. A Poisson/Gaussian mixture model was fit to the distribution of all *k*-mer multiplicities in order to model the process of generating incorrect (Poisson) and correct *k*-mers (Gaussian). To show the fit of the model, the *k*-mer multiplicities were generated according to the estimated parameters $\lambda = 0.95$, $\mu = 25$, and $\sigma = 9.38$ with a mixing parameter $w = 0.95$. One may find the multiplicity *m* with good separation between correct and incorrect *k*-mers by estimating the first local minimum from the distribution of *k*-mer counts or the minimum of the sum of the probabilities of the mixture model, a more smooth distribution. For multiplicity threshold $m = 5$, only 0.6% of correct 20-mers have multiplicity <5 and only 0.3% of incorrect 20-mers have multiplicity ≥5.

removing bulges, erroneous sources/sink edges, and chimeric reads as described in Chaisson and Pevzer (2008).

In high-coverage projects, fixed reads provide sufficient coverage across the genome to create a repeat graph that encodes the entire genome. However, many sequencing projects still contain low-coverage regions (often due to sampling bias). Since the error correction step relies upon the redundancy of *k*-mers in reads, the reads in regions of low coverage may be discarded at this step, causing fragmentation of the assembled contigs. We found that the unfixable partition often contains many reads from the low-coverage regions, and their exclusion from the assembly causes fragmentation of the repeat graph. In such cases it may be necessary to use what we call "Second Chance Assembly"—an assembly of all reads discarded during error correction (see Supplemental material).

Simplifying the repeat graph by transforming mate-pairs into mate-reads

Since the initial proposal to use mate-paired reads for shotgun sequencing (Weber and Myers 1997), mate-paired reads have been considered essential to de novo sequencing. Although in the past the short-read mate-paired data have not been available, various next-generation sequencing vendors have recently released modules for high-throughput production of mate-pairs. EULER-USR utilizes the mate-paired reads by modifying the EULER-DB approach (Pevzner et al. 2001) for incorporating mate-pairs into the Eulerian assembly framework.

When mate-pairs are available, the input to the fragment assembly is a set of mate-pairs: "read$_{start}$-GAP of length *d*-read$_{end}$." The key idea of EULER-DB (Pevzner and Tang 2001) is using the repeat graph to transform the mate-pairs "read$_{start}$-GAP of length *d*-read$_{end}$" into mate-reads "read$_{start}$-SEQUENCE of length *d*-read$_{end}$" by filling in the gap of length *d* with an appropriate path in the repeat graph. As a result, EULER-DB has an ability to generate contiguous long reads of length $2 \cdot l + d$ from short mate-paired reads of length *l*. These long mate-reads are subjected to the traditional Eulerian assembly afterward. Each mate-read corresponds to a mate-path between the mate-paired reads mapped to the repeat graph. In reality, the gap length is not fixed, but varies from $d - \delta$ to $d + \delta$.

While EULER-DB worked well for traditional Sanger sequencing (Pevzner et al. 2004), it needs to be modified for short reads. The key parameter of EULER-DB is the efficiency, the percentage of mate-pairs successfully transformed into mate-reads. This transformation is trivial if there is a single path of length $\approx d$ between read$_{start}$ and read$_{end}$ in the repeat graph, as the path is simply used to fill the gap between the reads. This was indeed the case for the overwhelming majority of Sanger reads making EULER-DB rather efficient. However, the repeat graphs for short reads are often very complex and in some cases there are multiple paths of length $\approx d$ between paired reads (Fig. 4). Pevzner and Tang (2001) described this problem and proposed an iterative approach to solve it (see Fig. 4A in Pevzner and Tang 2001). EULER-SR (Chaisson and Pevzer 2008), Velvet (Zerbino and Birney 2008), and ALLPATHS (Butler et al. 2008) all implement the idea of filling the gap between mate-pairs using the repeat graph and apply it to simulated data (e.g., cf. Fig. 4B in Pevzner et al. 2001 and Fig. 5C in Butler et al. 2008). Below, we apply EULER-DB to real Illumina mate-paired reads and show how to extend EULER-DB to analyze complex repeat graph characteristic for short-read sequencing.

The "Breadcrumb" and "All paths definition" procedures in Velvet and ALLPATHS are both aimed at filling up the gap between the mate-paired reads in the repeat graph. For most mate-pairs in *E. coli* there is only one path between mate-paired reads. In such cases, the paths found by the Breadcrumb and All paths definition methods are equivalent to EULER-DB. However, the remaining mate-pairs are important for resolving complex tangles and Velvet and ALLPATHS describe different approaches to addressing this challenge. Below we describe how a simple extension of EULER-DB addresses this problem.

When there are multiple paths in the repeat graph between a mate-pair (read$_{start}$, read$_{end}$), we may choose a path with maximum support from mate-pairs. Figure 4 illustrates the situation when there are many ways to transform the mate-pair (read$_{start}$, read$_{end}$) into a mate-read using one of the paths between them. Let edge e$_{start}$(e$_{end}$) be the edge where read$_{start}$ begins (read$_{end}$ ends). A mate-pair supports a path *P* between e$_{start}$ and e$_{end}$ if one of the reads is in either e$_{start}$ or e$_{end}$, and the other read is in an edge in *P*.

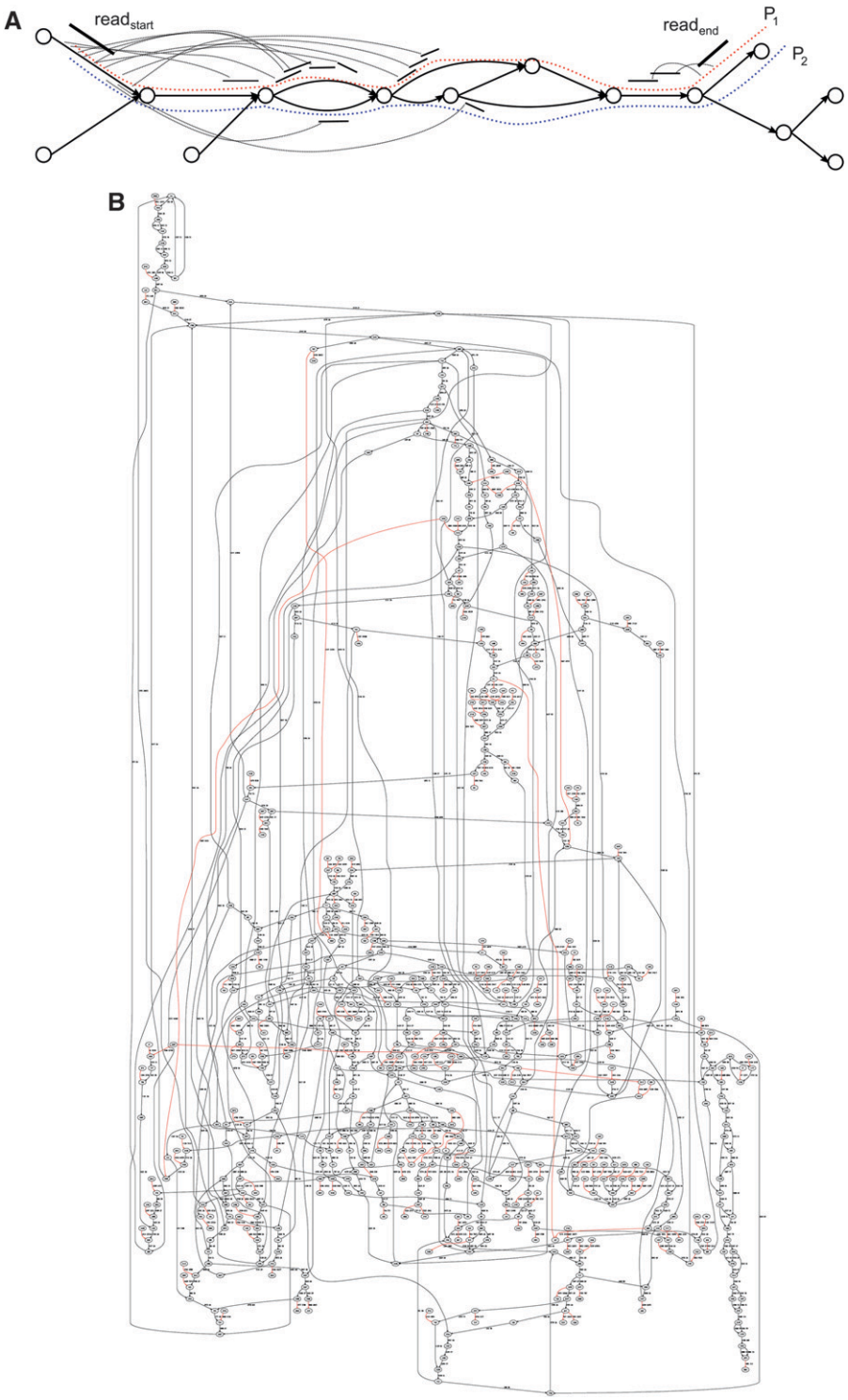

Figure 4. (*A*) A fragment of a made-up repeat graph formed by three divergent copies of a repeat. There are many possible paths from read$_{start}$ to read$_{end}$. To transform the mate-pair, read$_{star}$ -GAP of length *d*-read$_{end}$ into a mate-read read$_{start}$-SEQUENCE of length *d*-read$_{end}$, we compute the support for every path between read$_{start}$ and read$_{end}$ and select a path with maximum support. In this example, the "red" path P_1 has greater support than the "blue" path P_2. (*B*) A fragment of the real repeat graph of *E. coli* (constructed from ECOLI data set) illustrating that transformation of mate-pairs into mate-reads may fail in some cases. Red edges represent unique (typically long) contigs, while black edges represent repeats.

The number of such mate-pairs for a path P is denoted *support*(e_{start}, e_{end}, P). The path P with the highest value of *support*(e_{start}, e_{end}, P) is used to create the mate-path $P^+ = (e_{start}, P, e_{end})$. Note that due to gaps in coverage there may be edges in the path that are not supported. Furthermore, errors in reads may create reads that support edges not on the optimal path. Therefore, we use a path with maximal *Support*(*P*) among all paths between read$_{start}$ and read$_{end}$ unless *Support*(*P*) falls below a *MinSupport* threshold. Further details of transforming mate-pairs and into mate-reads are given in the Supplemental material.

Assembling error-prone reads (error correction by threading)

Each read corresponds to a unique read path in the de Bruijn graph representing the sequence of the read. Since the repeat graph approximates the de Bruijn graph, a similar argument applies to the read-paths in the repeat graph. A read may be mapped to the de Bruijn graph by aligning it to a closest subpath in the graph. Since the de Bruijn graph is built on read prefixes, the path corresponding to every read prefix (prefix path) is known, thus facilitating the read mapping. We may find the path that the entire read maps to by searching all subpaths continuing from the known prefix path, a process we refer to as threading reads through a graph.

In the case that threading is invoked, the EULER-USR(*k, m, l*) algorithm proceeds in five steps. The user has a choice of either specifying all three parameters: *k*, *m*, and *l*, or specifying a single parameter *k*. In the latter case, EULER-USR selects the suitable parameters *m* and *l* automatically.

1. Detecting accurate read prefixes and correcting errors within them using frequent *k*-mers (multiplicity *m* and higher). This operation generates the set of extremely accurate (nearly error-free) read prefixes.
2. Constructing the repeat graph on error-corrected prefixes using *k*-mers. This operation generates the set of *k-mer* contigs.
3. Threading entire reads through the repeat graph to extend the effective read length. This operation generates the set of accurate threaded reads.
4. Constructing the repeat graph on threaded reads and generating *l*-mer contigs using *l*-mers ($l > k$).
5. Simplifying the repeat graph by transforming mate-pairs into mate-reads.

Once the repeat graph has been constructed on the (accurate) read prefixes, we attempt to map every fixed read to the graph. However, while mapping of the (accurate) read prefixes is well-defined, mapping of (inaccurate) read suffixes is ambiguous. EULER-USR utilizes the repeat graph to correct errors in read suffixes.[6]

Every read *prefix*suffix* not discarded by error correction is represented as a concatenation of its prefix and suffix. Since the genome is a Eulerian traversal of its repeat graph, all substrings of the genome map to paths in the repeat graph. While the accurate prefix may be uniquely mapped to a path *path*(*prefix*) in the repeat graph, it is not clear how to map the entire read *prefix*suffix* since the suffix is inaccurate. We argue that to map *prefix*suffix*, one has to choose one of the extensions of *path*(*prefix*) among all paths of length n (read length) that begin with *path*(*prefix*). We denote the set of such paths as P and argue that a path in P with the minimum edit distance to the read represents the "best" mapping of the read *prefix*suffix* to the repeat graph. While in many cases such a *thread-path path*(*prefix*suffix*) may be used to correct the read *prefix*suffix*, it has to be done with caution (see below).

If P has a single edge (Fig. 5A), we correct the read with the sequence on the edge. However, these reads may not be used to resolve repeats and thus are vestigial in terms of improving the assembly. If P has multiple paths (Fig. 5B), we rank paths $P_1, P_2, \ldots$ in P in the increasing order of their Hamming distances $dist(P_1)$, $dist(P_2), \ldots$ to the read *prefix*suffix*. While it is tempting to choose the "optimal" path P_1 for correcting errors in the read *prefix*suffix*, it has to be done with caution. The problem is that the sequencing errors in the inaccurate suffix may transform it into an alternative string that maps to a "wrong" path in the repeat graph. We therefore check that (1) the optimal path P_1 is sufficiently similar to *prefix*suffix*, and (2) the second best path P_2 is sufficiently dissimilar from *prefix*suffix*. To check these conditions, we use P a parameter, f the expected error rate in read suffixes, and classify a path P as similar to the read if $dist(P) \leq f \cdot |suffix|$, and dissimilar otherwise. If both conditions (1) and (2) are satisfied, we use P_1 for correcting the read *prefix*suffix*, otherwise we iteratively trim before the position of the last difference, and rethread the read until both of these conditions are satisfied.

To obtain the final assembly that takes advantage of the longer threaded reads, we build the repeat graph on l-mers for $l > k$ (where k was used to build the original repeat graph), so that repeats of length l and shorter are resolved. The tradeoff between the k-mer size and repeat resolution is that edges from the repeat graph G of the genome (constructed on k-mers) will be split in the repeat graph of reads if there is a gap longer than $n - k$ between read start positions, where n is the read length. When $l > k$, the repeat graph of reads constructed on l-mers will be more fragmented than the repeat of reads constructed on k-mers. We prevent fragmentation of the repeat graph of reads constructed on l-mers by creating artificial reads $x \circ y$ for every pair of adjacent edges (contigs) x and y in the repeat graph of reads constructed on k-mers (k-mer contigs). One can either set the maximal $l = n - 2$ or empirically chose l to minimize fragmentation of the graph.

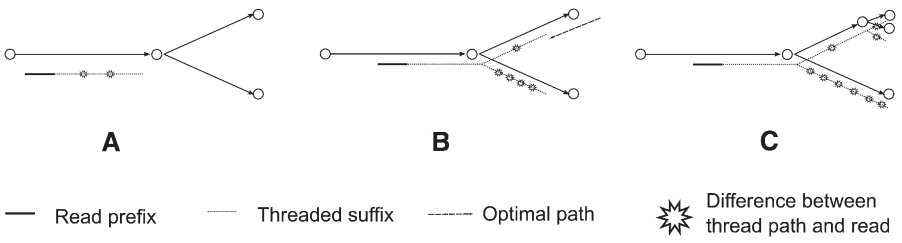

Figure 5. Mapping reads to the paths in the repeat graph. (*A*) A read maps to a single edge. (*B*) A read maps to two paths, and the closest one is chosen. (*C*) A read may be mapped to two similar paths implying that trimming is required.

Results

Data sets

We used the following data sets to benchmark EULER-USR and compare it to Velvet (Zerbino and Birney 2008), the most accurate among the recently published short-read assemblers.

- ECOLI. A set of 29.8 million paired Illumina reads from the ≈4.6 Mb *E. coli* genome (227× coverage). For benchmarking we randomly selected 10 million reads from this data set. Mapping of reads to the *E. coli* genome revealed that 87% are error-free. The separation between mate-pairs is 145 ± 25 bp.
- BAC50 and BAC35. A set of Illumina reads from an ≈170 Kb human BAC generated at the Joseph Ecker laboratory at the Salk Institute.[7] This BAC was sequenced over several runs, allowing us to generate an error profile that was not biased to a single run. A total of 2 million 35–50 base reads were generated for this BAC, resulting in 500× coverage and allowing us to choose an appropriate subset for a typical coverage benchmark. We randomly selected reads resulting in 50× coverage by 50-nt-long reads. Mapping these reads from the BAC to the reference sequence revealed that 20% are error free and 12% have only 1 error. Furthermore, 84% are error-free in the first 30 bases. While EULER-USR is designed to work with longer reads, the Velvet assembler is optimized for 35-base-long reads, and the performance deteriorates for longer reads. In order to make a fair comparison with Velvet we created a 35-nt read data set BAC35 by truncating 50-nt reads from BAC50 to 35 nt, resulting in 35× coverage of the BAC.
- simBAC100 and simECOLI100. A set of simulated 100 base reads from the human BAC. This set was generated to check whether EULER-USR can support extending sequencing reactions well beyond their prime. We simulated reads by mapping all 2 million reads from the data set generated at the Salk Institute to the BAC, extending them up to 100 bases, and simulating random errors. We further selected a set of resulting 100-base reads so that the coverage was 200× (to ensure that results were not biased by gaps in coverage). Errors in the resulting 100-base-long reads were simulated using a 1% error rate for the first 35 bases and 20% error rate for the remaining 65 bases. This error profile leads to a challenging assembly problem without attempting to model reads' characteristics for a particular technology. Our method for correcting errors using a repeat graph requires that the entire genome is covered by the high-quality prefixes of reads. The simBAC100 data set includes as many reads as the BAC50 data set to ensure that the entire genome is represented in the repeat graph constructed on 35-base read prefixes. The simECOLI100 is the set of simulated 100-base reads from *E. coli* (100× coverage). Errors were added to

[6]Error-corrected read suffixes only contribute to enlarging the assembled contigs and do not contribute to base calling.

[7]This BAC has a repeat content representative of the rest of the human genome.

Table 1. A comparison of assemblies of *E. coli* 35-base Illumina reads, unpaired and mate-paired

Assembly	N50	Length (# contigs) >20,000 nt	Length (# contigs) >5000 nt	Length (#contigs) >1000 nt
REPEAT-GRAPH(30)	22,173	2,432,772 (69)	4,232,578 (237)	4,484,685 (331)
EULER-USR unpaired	20,096	2,233,252 (68)	4,212,353 (249)	4,490,810 (355)
VELVET unpaired	16,424	1,953,255 (59)	4,068,326 (262)	4,484,065 (416)
EULER-USR mate-pairs	62,015	4,207,753 (72)	4,481,764 (96)	4,524,074 (113)
VELVET mate-pairs	45,427	3,800,552 (79)	4,419,542 (131)	4,507,932 (167)

N50: The size of the contig such that 50% of the assembly is contained in contigs of size N50 or greater. Length (# contigs) (>20,000 nt), total length of all contigs longer than 20,000 and total number of such contigs. We found that Velvet produces optimal results when run as Velvet(27,5).

the reads according to the same 1%–20% error profile used for simBAC100

- simBAC35. A set of 35-base read prefixes from simBAC100. Because the simBAC100 data set uses simulated reads, it is not directly comparable to the assemblies on real reads. Instead, to perform proper comparison for the effect of threading reads, we compare the assembly on 100-base (inaccurate) simulated reads to the assembly on the 35-base (accurate) simulated reads.

Benchmarking

We compared five recently published de novo short-read assemblers aimed at short reads: SSAKE (Warren et al. 2007), SHARCGS (Dohm et al. 2007), VCAKE (Jeck et al. 2007), Edena (Hernandez et al. 2008), and Velvet (Zerbino and Birney 2008). Of all assemblers, Velvet performed the best in terms of N50 contig size.

SSAKE is designed for assembling error-free reads, and SHARCGS is designed for reads with an error rate below 0.05%. Running these two methods on Illumina reads resulted in filtering of all reads on a preprocessing step or (if preprocessing is turned off) in an inferior assembly quality. VCAKE is an improvement of SSAKE and is able to assemble reads with higher error rate. For the BAC35 data set, VCAKE, Velvet, Edena, and EULER-USR produced similar assemblies. However, the quality of assemblies generated by VCAKE or Edena deteriorate with increasing the read length (BAC50 data set), producing more fragmented assemblies than Velvet. Furthermore, the assemblies for the ECOLI data set by both Velvet and EULER-USR resulted in doubling the N50 contig size as compared with VCAKE or Edena. We therefore decided to limit the detailed benchmarking to Velvet only.

A goal in the Eulerian approach is to construct the repeat graph G_k(Reads) on the set of Reads that best approximates the "ideal" repeat graph G_k(Genome) of the Genome (Chaisson and Pevzer 2008) (denoted as REPEAT-GRAPH[*k*] in the follow-up Tables and Figs.), as every Eulerian path in the repeat graph corresponds to a possible solution of the fragment assembly problem. However, due to fragmentation, the REPEAT-GRAPH statistics are misleading, because the longer the contigs are, the more likely they are to be fragmented by low-coverage regions (note that fragmentation of long contigs leads to a quick deterioration of the N50 size). Therefore, uneven distribution of reads over the genome may turn approximating G_k(Genome) into an unattainable goal. To set up a more realistic goal, we transform the set Reads into the error-free set PerfectReads (by substituting every read with the sequence of the genome it maps to). In the absence of mate-pairs, the repeat graph G_k(PerfectReads) constructed on this set of reads represents the best assembly our assembler aims for while assembling the real reads (referred to as OPTIMAL-ASSEMBLY in the follow-up Tables and Figures).

How are assemblies improved by mate-paired reads?

To benchmark EULER-USR and Velvet on the ECOLI data set, we first evaluated assembly with unpaired reads in order to later gage the effect of mate-pairs. Both the EULER-USR and Velvet (k = 27) assemblies were close to the theoretically optimal assembly (Table 1; Fig. 6) with similar N50 sizes (20 K for EULER-USR and 16 K for Velvet) and no misassemblies. The contigs longer than 500 bases in EULER-USR assembly (those likely to be nonrepetitive) contain six mismatches, three insertions, and one deletion (30 mismatches, three insertions, and three deletions in the Velvet assembly).[8]

Table 1 and Figure 6 compare EULER-USR and Velvet and illustrate that mate-pairs significantly improve the assemblies. The N50 length for *E. coli* assembly increases from 16 to 45 K for Velvet and from 19 to 62 K for EULER-USR. EULER-USR generated 127 contigs longer than 1000 bp, which is comparable to the typical number of contigs resulting from prefinished Sanger assembly of bacterial genomes (see Pevzner et al. 2004 and http://nbcr.sdsc.edu/euler/benchmarking/bact.html). Only two (relatively short) contigs produced by EULER-USR were misassembled. An

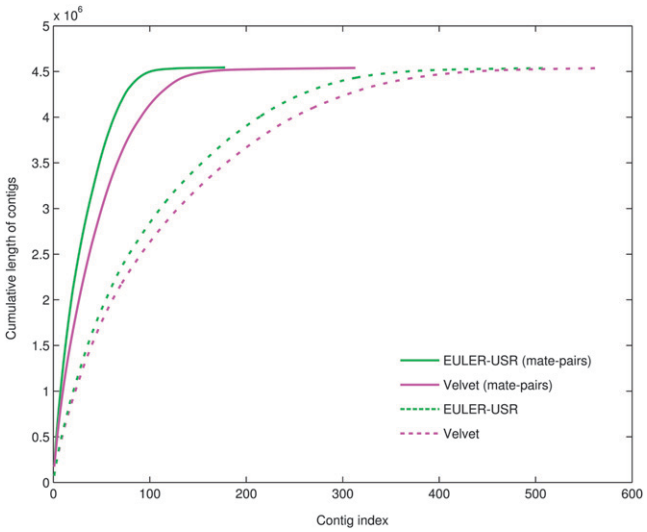

Figure 6. Comparison of EULER-USR and Velvet using both paired reads and the same reads with mate-pair information removed (ECOLI data set). The contigs are ordered in the decreasing order of sizes and the cumulative size of *x* longest contigs are shown on the *y*-axis.

[8]This analysis underestimates the base-calling errors by limiting them to long contigs and thus avoiding the most difficult repeated regions. Nevertheless, the base-calling accuracy appears to be comparable or even better than the accuracy of high-coverage Sanger sequencing.

Table 2. Error rate (per read base) and average length of reads on different stages of the EULER-USR threading algorithm

	Original reads		SA corrected reads			Threaded reads after graph correction	
Data set	Length	Error rate (%)	Average length	Error rate (%)	Retained reads (%)	Average length	Average rate (%)
BAC35	35	0.92	34.9	0.01	91.3	34.9	0.004
BAC50	50	4.36	46.7	0.04	88.6	49.3	0.049
simBAC100	100	13.3	46.6	0.07	98.0	94.5	0.050
simECOLI100	100	12.6	50.5	0.003	99.6	98.8	0.017

The error rate is computed by mapping reads to the genome. We compute the length and error rate for the original reads, reads corrected by Spectral Alignment that are retained after graph correction, and finally, after threading. The increased error rate after threading is due to threading reads through their consensus sequence in the repeat graph (rather than de Bruijn graph).

alignment of the assemblies to the *E. coli* genome is shown in the Supplemental material.

How are assemblies improved by read threading?

When estimating input parameters for EULER-USR, our mixture model suggested the multiplicity threshold $m = 5$ for the BAC50 data set with k-mer size 20. The accuracy of error correction was evaluated by mapping error-corrected reads to the BAC (Table 2). From the original sets of reads, 91.3% of the BAC35 data set, 88.6% of the BAC50 data set, and 98.0% of the simBAC100 data set were retained after error correction based on Spectral Alignment. During graph correction, the removal of certain edges from the graph truncates the reads that map to these edges, further shortening the average read length. While most reads in this data set were trimmed only by a few nucleotides, others become rather short and need to be extended with threading as described above.

Table 3 presents the statistics of the N50 contig size, as well as the cumulative contig size for various data sets (reported for contigs longer than 1000, 500, and 100 bases). Since the N50 statistic is limited, we show the differences in assemblies by plotting the cumulative length of contigs ordered by size (Fig. 7). Figure 7 shows how longer threaded reads improve the assembly quality for both real and simulated reads.[9] Figure 7 illustrates that while Velvet and EULER-USR show similar results for the BAC35 data set (no read threading), the EULER-USR assembly improves for BAC50 and simBAC100 data sets (due to its ability to utilize the error-prone reads), while Velvet assembly hardly changes. Indeed, even with a modest increase in read length from 35 to 50 nt, read threading increases N50 contig size by 13% and the total length of long contigs (longer than 1000 nt) by 22%. Figure 8 shows how assembly improves with increase in read coverage when assembling the *E. coli* genome and leads to a conclusion that the coverage increase beyond 55× results only in a modest increase in assembly quality.

When the BAC50 data set is assembled without threading reads, the N50 contig size is 1752, with a net assembly size of 171,301. Read threading only improves the quality of assembly by correcting reads that pass through three or more edges (most reads map to one or two edges in the repeat graph). Table 4 shows the number of reads that are correctly fixed with threading and how many edges they are threaded through for the BAC50 data set (see Supplemental material for analysis of simBAC100 data set). Although a very small fraction of reads are threaded through more than three edges, they improve the quality of assembly.[10]

Since bacterial genomes have a compact gene structure, we analyzed how many genes are captured within contigs constructed by various programs. Even though the repeat graph of *E. coli* is fragmented into many contigs (over 500 for REPEAT-GRAPH[30]), many contigs are long, and contain multiple genes. We mapped the contigs of REPEAT-GRAPH, EULER-USR, and Velvet assemblies to the genome, and counted the number of genes that contained entirely within a contig. Table 5 illustrates that contigs produced by Velvet and EULER-USR capture a large number of bacterial genes, thus enabling various applications. For example, one can perform MS/MS proteomics analysis of bacterial genomes with Illumina contigs almost as efficiently as with completed genomes (Gupta et al. 2007).

The *E. coli* genome contains relatively few repetitive elements compared with the human genome, and so the longer reads should be able to resolve more repeats in *E. coli* than in the human BAC. In the simECOLI100 data set, the usable read length was increased from an average of 50.5 to 98.8 nt (Table 2) by threading. When we compare the assembly on the simulated read prefixes to the threaded reads, the N50 contig size more than doubles to ≈45 K. This indicates that the announced increase in Illumina read length to 100 nt (planned for 2009) will lead to significant improvement in assembly in the case of unpaired reads. While it is an important improvement in applications like single-cell sequencing (where the mate-paired protocols are not available yet), it remains unclear whether the read length (as opposed to span) matters in the case of mate-paired reads (i.e., would increasing the length of mate-paired reads from 35 to 100 nt lead to significant improvements in assembly if the span remains fixed?).

Does the read length matter?

The availability of two methods to resolve repeats (mate-pairs and threading) brings the question as to whether they may be used in conjunction to further improve assemblies. To test this, we simulated mate-pairs with the span 300 ± 30 nt in the genomes of *E. coli* and *S. cerevisiae*. The read length r was fixed in each simulated data set, with a minimum read length of 25 nt and maximum length 100 nt. The goal was to evaluate whether the quality of assembly (e.g., N50(r), N50 length for reads of length r) with longer reads improves as compared to the quality of assembly with shorter reads (e.g., whether 100 nt mate-paired reads result in a better assembly than 35 nt reads). We define the read-length barrier as the read length after which the quality of assembly does not significantly improve (e.g., N50 does not increase by >5%).

[9]In some cases the statistics for EULER-USR is slightly better than for OPTIMAL-ASSEMBLY due to subtle differences in contig reporting.

[10]For BAC50 data set, EULER-USR has 20 mismatches and two insertions, a higher error rate as compared with ECOLI data set.

Table 3. Assembly statistics of various data sets of Illumina reads

Assembly method/data set	N50	Length (# contigs) >1000	Length (# contigs) >500	Length (# contigs) >100
BAC35				
REPEAT-GRAPH(25)	1869	97,946 (34)	126,774 (74)	153,378 (194)
OPTIMAL-ASSEMBLY(25)	1609	92,758 (39)	119,773 (78)	153,348 (225)
EULER-USR(20,2,25)	1786	89,905 (39)	118,576 (80)	147,227 (210)
VELVET(21,5)	1428	87,551 (43)	113,522 (80)	138,554 (173)
BAC50				
REPEAT-GRAPH(40)	4168	143,224 (39)	160,679 (64)	175,246 (128)
OPTIMAL-ASSEMBLY(40)	2023	129,392 (55)	152,551 (87)	176,491 (193)
EULER-USR(20,5,40)	2022	119,489 (45)	148,319 (86)	171,082 (164)
VELVET(31,5)	1381	84,292 (39)	112,886 (78)	139,176 (169)
simBAC35				
REPEAT-GRAPH(25)	1869	97,946 (34)	126,774 (74)	153,378 (194)
OPTIMAL-ASSEMBLY(25)	1847	95,180 (35)	159,359 (85)	175,305 (242)
EULER-USR(20,5,25)	1818	98,187 (39)	129,671 (85)	162,109 (242)
VELVET(21,5)	1844	97,174 (36)	122,816 (73)	144,578 (153)
simBAC100				
REPEAT-GRAPH(50)	7163	167,112 (31)	172,990 (39)	175,444 (53)
OPTIMAL-ASSEMBLY(50)	3971	162,129 (47)	169,794 (58)	176,908 (94)
EULER-USR(20,5,50)	2639	140,718 (47)	163,244 (80)	175,900 (135)
VELVET(31,5)	695	36,796 (22)	77,557 (80)	120,147 (241)
simECOLI100				
REPEAT-GRAPH(50)	59,656	4,519,592 (140)	4,528,404 (152)	4,583,803 (533)
EULER-USR(20,10,50)	44,710	4,519,083 (182)	4,531,837 (200)	4,562,551 (366)

(N50) The size of the contig such that 50% of the assembly is contained in contigs of size N50 or greater. Length (>1000), Length (>500), and Length (>100): the total length of all contigs longer than 1000, 500, and 100 nt, respectively. For Velvet (*k*-mer size, coverage) we found that the coverage cutoff $t = 5$ maximizes the assembly quality. The effect of threading reads on assembly quality may be seen by comparing simBAC35 and simBAC100. The rows describing REPEAT-GRAPH(50) and OPTIMALASSEMBLY(50) are identical, since the reads cover the entire BAC in simBAC100 data set. In all tests there was a single misassembly (in simECOLI100 data set).

The error-free mate-paired reads were simulated starting at every genomic position and each simulated data set with reads of length r were assembled with EULER-USR (*k*-mer size 24). To evaluate the quality of assemblies, we used N50 contig size and efficiency (the percentage of mate-pairs transformed into mate-reads). The efficiency should be analyzed with caution because only mate-pairs spanning multiple edges in the repeat graph contribute to improving the assembly (a similar effect is illustrated in Table 4 for single reads). In the *E. coli* assembly of Illumina reads, this was only 3.9% of all mate-pairs. As a result, for the simulated *E. coli* assembly, the efficiency is rather high for all read lengths (varying from 97.8% for $r = 25$ to 99.0% for $r = 55$, and to 99.2% for $r = 100$). However, even a small increase in efficiency translates into significant increase in N50 statistics (varying from ≈40 K for $r = 25$ to ≈60 K for $r = 55$). Therefore, small increases in efficiency may reflect a very significant increase in the number of "useful" mate-pairs, i.e., mate-pairs that improve the assembly. To better gage the contribution of such "useful" mate-pairs, we introduce the relative efficiency, the percentage of useful mate-pairs transformed into mate-reads (we call a mate-pair useful if its reads

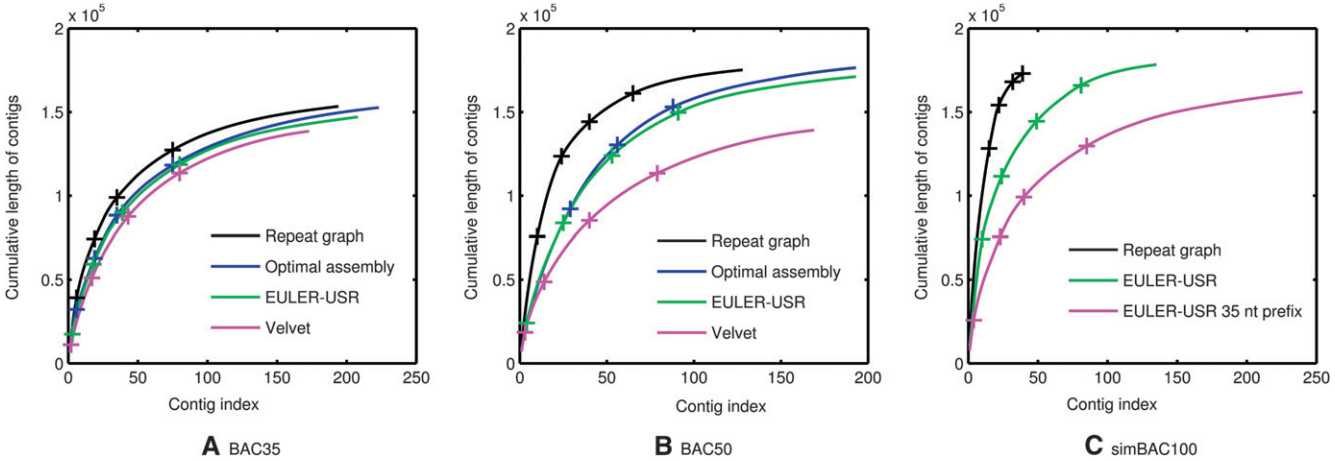

Figure 7. Comparison of EULER-USR (threading) and Velvet. In each plot, the contigs are ordered in the decreasing order of size and the cumulative size of x longest contigs are shown on the y-axis (only contigs longer than 100 bases are shown). See Table 3 for the choice of parameters of all programs in these plots. For all assemblies of the BAC, the locations of the contigs closest to lengths 5000, 2000, 1000, and 500 bases are shown with a "+" mark.

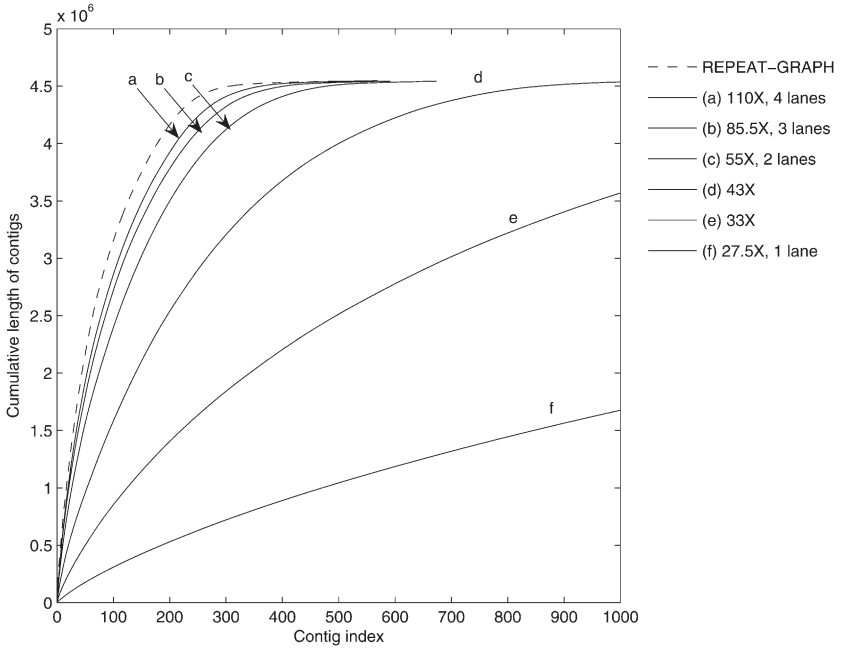

Figure 8. Statistics of assembly for various read coverage (*E. coli* genome). The cumulative length of contigs in order of decreasing length is shown for the 1000 longest contigs. The cumulative length of contigs of the repeat graph on the genome is shown as a dashed line.

reside on different edges). For the yeast data set, the relative efficiency varies from 61% for $r = 25$ to 80% for $r = 55$ (the maximum value among all read lengths is 81%). These results indicate that for the yeast data set, efficiency hardly changes after the read length exceeds ≈60 nt.

Our attempt to answer the question "Does the read length matter?" is limited in many aspects (e.g., reads were simulated error-free, and coverage was perfectly uniform) and it only answers the question of whether the read length matters for EULER-USR assemblies (rather than for a theoretically optimal assembly with mate-pairs).[11] However, it reveals that for the *E. coli*, the assembly hardly improves after the read length exceeds 35 nt (efficiency = 98.7%, N50 contig size ≈ 60 Kb). The assembly deteriorates when the read length decreases from 35 to 25, indicating that the read-length barrier for *E. coli* (with the chosen simulation parameters) is ≈35 nt.[12]

For the *S. cerevisiae* genome, the assembly quality only slightly improves after the read length exceeds 60 nt (N50 is ≈70 K for $r = 60$ but drops to ≈62 K at $r = 45$ and to ≈41 K at $r = 25$). It indicates that the read-length barrier for *S. cerevisiae* (with chosen simulation parameters) is ≈60 nt.

Discussion

The recent addition of mate-paired reads to the arsenal of short-read technologies opened the possibility of assembling complex genomes for a fraction of the cost of the traditional Sanger sequencing. We demonstrated that the Eulerian approach is well suited for assembling mate-paired short reads by transforming mate-pairs into mate-reads using repeat graph. We further complemented the approach from Pevzner et al. (2001) by selecting the most "supported" mate-reads to resolve some difficult cases when a mate-pair may be transformed into multiple mate-reads.

In addition to incorporating mate-pairs into fragment assembly, we also show that the conventional wisdom of "read trimming" may be substituted by threading to correct error-prone read tails. We demonstrate that if a sequencing technique "suffers" from quality degradation along the length of a read, it may still be used effectively in de novo assembly. Despite the fact that short-read assemblies are rather fragmented, we demonstrate that most bacterial genes map to single contigs, thus enabling gene discovery and annotation of bacterial genomes.

The Eulerian approach models the error-prone suffixes of the reads as short edges to vertices of out-degree zero. All recently developed short-read assemblers remove such edges from the graph (e.g., via the erosion procedure in Pevzner et al. 2001), thus essentially discarding information contained in the error-prone read suffixes. Therefore, even if the reads are not explicitly trimmed, they are implicitly trimmed after the de Bruijn graph is constructed (e.g., using the "clipping" procedure in Velvet [Zerbino and Birney 2008] or "removal of hanging-ends" procedure in ALLPATHS [Butler et al. 2008]). EULER-USR differs from these approaches by utilizing information in the error-prone read prefixes.

Our study on the use of mate-paired reads in conjunction with read threading revealed that there exists some synergy between these two approaches when the read length remains below the read-length barrier. While mate-pairs represent the major factor in improving the assembly quality, read threading contributes to further improvements in assembly. The next challenge for short-read technologies is to assemble larger and more complex genomes. The ability to exploit any information possible to

Table 4. The results of read threading for BAC50 data set

No. reads	Total	Reads spanning one edge	Reads spanning two edges	Read Spanning >2 edges	Average read length (after threading)
Correct/correct	100,942	95,781	2161	2550	50
Correct/incorrect	207	174	27	6	50
Incorrect/correct	55,515	52,408	1464	1643	42
Incorrect/incorrect	347	254	33	60	45

Reads are classified into four categories: correct/correct (if threading does not change a correct read), correct/incorrect (if threading turns a correct read into incorrect), incorrect/correct (if threading turns an incorrect read into correct), and incorrect/incorrect (if threading turns an incorrect read into an incorrect read). The table classifies reads in each of these four categories depending on how many edges in the repeat graph they span.

[11]The theoretically optimal algorithms for assembling mate-paired reads remain unknown even for error-free reads and fixed distance between mate-pairs (Medvedev et al. 2007).

[12]We found that assemblies of mate-pairs with average span $d \pm \sigma$ may be sensitive to the parameter even σ for the same d. For example, simulated assemblies with error-free reads may have lower quality than the real assemblies with the same d but different σ.

Table 5. Mapping of the 4136 genes from *E. coli* into theoretical repeat graph and repeat graph constructed by EULER-USR and Velvet for the ECOLI data set

Method	REPEAT-GRAPH	EULER-USR	VELVET
# Complete genes	3937 (95.2%)	3956 (95.7%)	3912 (94.6%)

See Table 1 for a description of the parameters. We show the number of genes that are contained entirely within the constructed contigs.

resolve repeats will become important when assemblers move to mammalian genomes.

Acknowledgments

This research was supported by NIH grant 1R21HG004130-01 and NSF grant EIA-0303622. We thank Dirk Evers, Klaus Maisinger, and Jacques Retief for insightful discussions about the Illumina technology, and to Xiaohua Huang and Eric Roller for many discussions on emerging next-generation sequencing technologies. We are grateful to Ronan O'Malley and Joseph Ecker for providing us with the BAC reads.

References

Barski, A., Cuddapah, S., Cui, K., Roh, T.Y., Schones, D.E., Wei, G., Chepelev, I., and Zhao, K. 2007. High-resolution profiling of histone methylations in the human genome. *Cell* **129:** 823–837.

Butler, J., MacCallum, I., Kleber, M., Shlyakhter, I.A., Belmonte, M.K., Lander, E.S., Nusbaum, C., and Jaffe, D.B. 2008. ALLPATHS: De novo assembly of whole-genome shotgun microreads. *Genome Res.* **18:** 810–820.

Chaisson, M.J. and Pevzner, P.A. 2008. Short read fragment assembly of bacterial genomes. *Genome Res.* **18:** 324–330.

Chaisson, M.J., Tang, H., and Pevzner, P.A. 2004. Fragment assembly with short reads. *Bioinformatics* **20:** 2067–2074.

Dohm, J.C., Lottaz, C., Borodina, T., and Himmelbauer, H. 2007. SHARCGS, a fast and highly accurate short-read assembly algorithm for de novo genomic sequencing. *Genome Res.* **17:** 1697–1706.

Gupta, N., Tanner, S., Jaitly, N., Adkins, J.N., Lipton, M., Edwards, R., Romine, M., Osterman, A., Bafna, V., Smith, R.D., et al. 2007. Whole proteome analysis of post-translational modifications: Applications of mass-spectrometry for proteogenomic annotation. *Genome Res.* **17:** 1362–1377.

Hernandez, D., Franois, P., Farinelli, L., Osteras, M., and Schrenzel, J. 2008. De novo bacterial genome sequencing: millions of very short reads assembled on a desktop computer. *Genome Res.* **18:** 802–809.

Hillier, L.W., Marth, G.T., Quinlan, A.R., Dooling, D., Fewell, G., Barnett, D., Fox, P., Glasscock, J.I., Hickenbotham, M., Huang, W., et al. 2008. Whole-genome sequencing and variant discovery in *C. elegans*. *Nat. Methods* **5:** 183–188.

Huang, X., Wang, J., Aluru, S., Yang, S., and Hillier, L. 2003. PCAP: A whole genome assembly program. *Genome Res.* **13:** 2164–2170.

Idury, R.M. and Waterman, M.S. 1995. A new algorithm for DNA sequence assembly. *J. Comput. Biol.* **2:** 291–306.

Jaffe, D.B., Butler, J., Gnerre, S., Mauceli, E., Lindblad-Toh, K., Mesirov, J.P., Zody, M.C., and Lander, E.S. 2003. Whole-genome sequence assembly for mammalian genomes: Arachne 2. *Genome Res.* **13:** 91–96.

Jeck, W.R., Reinhardt, J.A., Baltrus, D.A., Hickenbotham, M.T., Magrini, V., Mardis, E.R., Dangl, J.L., and Jones, C.D. 2007. Extending assembly of short DNA sequences to handle error. *Bioinformatics* **23:** 2942–2944.

Margulies, M. and Egholm, M. 2005. Genome sequencing in microfabricated high-density picolitre reactors. *Nature* **437:** 326–327.

Medvedev, P., Georgiou, K., Myers, G., and Brudno, M. 2007. Computability of models for sequence assembly. In *Proceedings of the Seventh International Workshop,* pp. 289–301. WABI, Philadelphia, PA.

Myers, E.W. 2005. The fragment assembly string graph. *Bioinformatics* Suppl 2: ii79–ii85. doi: 10.1093/bioinformatics/bti7114.

Ng, P., Tan, J.J., Ooi, H.S., Lee, Y.L., Chiu, K.P., Fullwood, M.J., Srinivasan, K.G., Perbost, C., Du, L., Sung, W.K., et al. 2006. Multiplex sequencing of paired-end ditags (MS-PET): A strategy for the ultra-high-throughput analysis of transcriptomes and genomes. *Nucleic Acids Res.* **34:** e84. doi: 10.1093/nar/gki444.

Pevzner, P.A. 1989. 1-Tuple DNA sequencing: Computer analysis. *J. Biomol. Struct. Dyn.* **7:** 63–73.

Pevzner, P.A. and Tang, H. 2001. Fragment assembly with double-barreled data. *Bioinformatics* **17:** S225–S233.

Pevzner, P.A., Tang, H., and Waterman, M.S. 2001. A Eulerian path approach to DNA fragment assembly. *Proc. Natl. Acad. Sci.* **98:** 9748–9753.

Pevzner, P.A., Tang, H., and Tesler, G. 2004. De novo repeat classification and fragment assembly. *Genome Res.* **14:** 1786–1796.

Schones, D.E. and Zhao, K. 2008. Genome-wide approaches to studying chromatin modifications. *Nat. Rev. Genet.* **9:** 179–191.

Tammi, M.T., Arner, E., Kindlund, E., and Andersson, B. 2003. Correcting errors in shotgun sequences. *Nucleic Acids Res.* **31:** 4663–4672.

Warren, R.L., Sutton, G.G., Jones, S.J.M., and Holt, R.A. 2007. Assembling millions of short DNA sequences using SSAKE. *Bioinformatics* **23:** 500–501.

Weber, J.L. and Myers, E.U. 1997. Human whole genome shotgun sequencing. *Genome Res.* **7:** 401–409.

Whiteford, N., Haslam, N., Weber, G., Prugel-Bennett, A., Essex, J.W., Roach, P.L., Bradley, M., and Neylon, C. 2005. An analysis of the feasibility of short read sequencing. *Nucleic Acids Res.* **33:** e171. doi: 10.1092/nar/gni171.

Zerbino, D.R. and Birney, E. 2008. Velvet: Algorithms for de novo short read assembly using de Bruijn graphs. *Genome Res.* **18:** 821–829.

Received March 26, 2008; accepted in revised form November 17, 2008.